AFRL-RH-WP-TR-2008-0041

LGIST: Learning Generalized Image Schemas for Transfer

Paul Cohen
Carole Beal

University of Southern California
Information Sciences Institute
4676 Admiralty Way, Suite 1001
Marina del Rey CA 90292

February 2008

Final Report for August 2005 to February 2008

Air Force Research Laboratory
Human Effectiveness Directorate
Warfighter Interface Division
Cognitive Systems Branch
Wright-Patterson AFB OH 45433-7022

NOTICE

AFRL-RH-WP-TR-2008-0041

//SIGNED//
BRIAN H. TSOU
Work Unit Manager
Cognitive Systems Branch

//SIGNED//
DANIEL G. GODDARD
Chief, Warfighter Interface Division
Human Effectiveness Directorate
Air Force Research Laboratory

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
February 2008	Final	August 2005 – February 2008

4.TITLE AND SUBTITLE
LGIST: Learning Generalized Image Schemas for Transfer

5a. CONTRACT NUMBER
FA8650-05-C-7266

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
63123F

6. AUTHOR(S)
Paul Cohen, Carole Beal

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER
DRPA0513

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University of Southern California
Information Sciences Institute
4676 Admiralty Way, Suite 1001
Marina del Rey CA 90292

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Materiel Command
Air Force Research Laboratory
Human Effectiveness Directorate
Warfighter Interface Division
Cognitive Systems Branch
Wright-Patterson AFB OH 45433-7022

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RHCS

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
AFRL-RH-WP-TR-2008-0041

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

88th ABW/PA cleared on 08 May 2008, WPAFB-08-3189.

14. ABSTRACT

This research was funded under a Thrust D award to develop algorithms for *gist memory*. Gists are abstractions that preserve the important causal events in activities or episodes, eliding details. The point of this research was to develop a new kind of representation, *image schemas*, and associated learning methods. The learning agent was called *Jean*. *Jean* is a model of early cognitive development based loosely on Jean Piaget's theory of sensori-motor and pre-operational thought. Like an infant, *Jean* repeatedly executes schemas, gradually transferring them to new situations and extending them as necessary to accommodate new experiences. We model this process of accommodation with the Experimental State Splitting (ESS) algorithm. ESS learns elementary action schemas, which comprise controllers and maps of the expected dynamics of executing controllers in different conditions. ESS also learns compositions of action schemas called gists. We present tests of the ESS algorithm in three transfer learning experiments, in which *Jean* transfers learned gists to new situations in a real time strategy military simulator.

15. SUBJECT TERMS Gists, Image Schemas, Algorithms, Experimental State Splitting (ESS), Learning Agent

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Brian H. Tsou
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR	30	19b. TELEPHONE NUMBER *(include area code)*

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. 239.18

THIS PAGE LEFT INTENTIONALLY BLANK

Table of Contents

List of Figures

List of Tables

1 Synopsis

We were funded under a Thrust D award to develop algorithms for *gist memory*. Gists are abstractions that preserve the important causal events in activities or episodes, eliding details. Tactical plans such as ambushing an enemy are gists: the essential causal elements are that one remains hidden from the enemy until an advantageous moment, and then falls upon him suddenly. We proposed to develop an algorithm to learn gists for tactical plans in the ISIS small-unit tactics simulator (ISIS is a military battlefield simulator based on a real-time strategy game). Results have been good (see Sec. 7.3 and [8]).

However, the point of our Biologically-Inspired Cognitive Architecture (BICA) study was less to learn military tactics than to develop a new kind of representation, *image schemas*, and associated learning methods. We had been interested in image schemas for several years as a possible foundation for cognitive development, and, while it is not the goal of this project to "build a baby," we found that the learning methods we designed for image schemas and gists are plausible learning methods for infants. Our learning agent, called Jean in honor of Jean Piaget and Jean Mandler, has a small set of image schemas that serve as the innate sensori-motor endowment, and learns as infants do, by repeatedly executing primitive actions and previously learned gists.

At this writing, the Jean project is a bundle of related and unfinished threads. Let us introduce them very briefly. We developed a representation language for spatial arrangements called the *Image Schema Language* (ISL, Sec. 5). It soon became clear that the image schemas that inspired us originally, from the Cognitive Linguistics literature, are mostly about static arrangements, whereas much commonsense knowledge is about dynamics. Gists, in particular, are abstractions of activities, so Jean would need a representation of activities. We wanted real dynamics, not the faux dynamics of the situation calculus, so we developed two representations of dynamics. The first, based on finite state machines, was the subject of several papers [18, 5, 32], but it was unsatisfactory for several reasons. More recently we have developed a representation of dynamics based on *maps* (Sec. 5.3), but we have no experimental results to report, yet. The *Experimental State Splitting* (ESS) algorithm learns action schemas, and also stitches them together to form gists from sequences of actions (Sec. 6.1). Jean recently got another algorithm, for learning word meanings (Sec. 8). This algorithm and ESS were developed and tested independently and in different environments — ESS in the ISIS military tactics simulator, the lexical acquisition algorithm in Jean's Room, a physics-based simulator in which Jean is instructed in English to do things with toy blocks. We need to integrate the various parts of Jean and test the resulting system in both environments.

We intend to team up with BBN Technologies and the Massachusetts Institute of Technology (MIT) Computer Science and Artificial Intelligence Laboratory (CSAIL) for Phase II of BICA. The MIT group based their architecture on control loops that are similar in many ways to what we call action schemas and gists. Jean's "architecture" is little more than a bundle of processes (see Sec. 4). It was not the point of our Thrust D effort to design an architecture, and the MIT team has one, whereas we have designed a new class of knowledge representation (image schemas and gists) and learning methods.

2 Setting the Context: Developmental AI

We view image-schematic representations and learning methods as developmental psychologists do, as plausible mechanisms for human cognitive development. Why might DARPA care about development? One reason is that image-schematic representations are inherently spatial and dynamical, and well-suited to representing and learning military tactics.

Another reason, which speaks more directly to modeling development, is that thousands of AI researchers have not been able to achieve in fifty years what a newborn child does in a single year. Cognitive development appears to be effortless, inevitable, and amazingly rapid, which leads many psychologists to believe that the human infant is born with a kind of "developmental potential energy." The infant architecture — its sensors, effectors, reflexes, preferences and mental processes — seems designed to change. In the right environment, this architecture transforms itself into the mind of a toddler, then into the concrete, literal mind of the elementary-school child, and, finally, into an adult abstract reasoner.

If we could figure out how the infant mind is built, then perhaps we could build AI systems with the same propensity to bootstrap and self-organize. There are good reasons to focus on infant and toddler intelligence: While efforts to build general adult intelligence have failed for want of commonsense knowledge, infants and toddlers acquire this knowledge effortlessly— it is what they *do*. While contemporary machine learning strangles in the grip of intellectual machismo, striving to solve harder problems in more sophisticated ways, infants and toddlers learn in very simple ways from enormously rich perceptual and social information. Nowadays, approximations to these sorts of information are available from physics-based simulators and web-based social mechanisms such as MMOGs and chat rooms. It is time to try to build an infant intelligence and take seriously the challenge of learning common sense and all that depends on it, including language.

3 Architectural Principles of Cognitive Development

The goal, then, is a bootstrapping, self-organizing cognitive architecture. The architectural principles of the infant mind are moderately well-understood:

Sensori-motor Activity First. The most important concept in the architecture of the infant mind goes by several names: Piaget and many others (including us) called it *schemas*, the MIT CSAIL group calls it *loops*. What roboticists call *controllers*, modern developmental psychologists call *dynamical systems*. All refer to the idea that the infant exercises sensori-motor behaviors of increasing coordination and complexity, and in the process learns a great deal about her physical world.

The first schemas are probably reflexes. The infant quickly achieves voluntary movement, although it is poorly coordinated and controlled. Coordinated behaviors, integrating sensing and action, goals and predictions, soon develop. We call these larger compositions *gists*. In Section 6.1 we describe an algorithm for learning gists.

Consider seeing a toy, reaching for it, grasping it and banging it on the table. This simple gist involves the coordinated activity of perception, memory, planning, motor schemas, expectation and prediction, and learning. Esther Thelen, in her pioneering studies of motor learning, discovered that loops such as coordinated reaching and grasping are not preprogrammed and do not develop in the same ways in all children [35]. Instead, because children have different physical capabilities, they must learn to coordinate them — a stronger child learns to attenuate her arm movements, a child with poorer visual acuity learns to move slightly more slowly. The child learns to assemble components of activity in ways that guarantee outcomes such as grasping a toy. Unstable loops, such as leaning too far forward and toppling, give way to stable loops that work in most situations.

Coordination and Mutual Bootstrapping. No aspect of cognition develops in isolation from the rest. Motor development depends on perceptual development; language development depends on conceptual and social development, and vice versa.

Perceptual Invariances and Affordances. The origins of commonsense knowledge are in the child's perceptual system. The visual system provides many examples: It innately makes a figure/ground distinction, computes size-constancy, responds to optic flow and looming, and roughly parses the visual scene into objects. Very early on, the child discriminates animate and inanimate motion (e.g., the way a human or dog vs. a ball move). The child learns quickly to correlate events across sensory modes (e.g., the sound and visual impressions of a ball bouncing).

Perception often gives rise to *invariances*, which are reliable and unchanging relationships that can be extracted from changing perceptual arrays. Kinetic depth, for instance, is the constant relationship between the distance between two objects and their rates of apparent movement when one moves one's head. Two-month-olds use kinetic depth (rather than binocular depth, perhaps because their visual acuity is poor) to judge distances between objects. Perhaps kinetic depth and other perceptual invariances are the genesis of a relational representational system.

The child quickly learns *affordances*, which are relationships between how things appear and what one can do. Rigid things with "handles" afford shaking, banging on surfaces, and so on. Any such thing will do — spoons, chopsticks, and rattles, but not balls, socks, and stuffed toys. The affordances of objects are the earliest commonsense knowledge.

Learning by Doing. The AI approach to commonsense knowledge is to axiomatize; the child's approach is to act. Young children learn most of what they know by doing and they are largely incapable of learning any other way. Tell a toddler that pillows don't bounce, and she will learn nothing. Drop a pillow a few times, and she's got it. Exercising a "reach and grasp" gist with different toys produces knowledge about rigidity and flexibility, texture and color, contingent action and effects (e.g., banging an object produces noise), and so on. Not until children are seven or eight *years* old can they learn from definitions (axioms), and, even then, they need concrete examples to make the concepts stick.

Memory and Semantics over Reasoning. Human reason is slow and shallow, but human memory is timely and semantically deep. The young child, especially, lacks logical reasoning skills and language, yet can recognize and predict, classify and associate, communicate and understand, and has goals and acts to achieve them. All this is accomplished with little of what adults call reasoning. Instead, the child relies on a memory system that appears designed to extract or at least index information semantically. Children acquire their concepts and ontologies young. During the first year, infants apparently distinguish living from non-living things, manufactured objects from natural objects, children from adults, males from females, and numerous other distinctions. They discriminate based on gender, animacy, emotional cues in speech, and many other abstract properties. Some distinctions are based on perceptual differences, others on functional differences, and the child quickly coordinates perceptual and functional classes (e.g., the class of "things to bang on the table" is recognizable for its perceptual features as well as what one can do with instances.)

Epigenetics. There is vigorous debate about how much newborns know about the physical world and whether their perceptual systems provide them with structured representations or with "blooming, buzzing confusion." (The methodology of experiments with infants, particularly habituation studies, is suspect.) It is safe to say that strong nativist and empiricist positions will probably not prove true, but instead, the infant has innate structure — certainly in its perceptual system — that determines the kinds of knowledge it develops.

4 Introducing Jean

Jean is an agent that models aspects of human cognitive development. Versions of Jean have been situated in three environments: A simple three-dimensional environment based on the Open Dynamics Engine [1], a small-unit tactics game called ISIS, and Jean's Room, a Shockwave simulation in which a robot-like Jean can interact with toy blocks. Development of Jean will continue primarily in Jean's Room.

Jean's architecture implements four processes, as shown in Figure 1. Calling on a store of image schemas, *perception* involves building at least one, and usually many, image-schematic representations of the current situation. *Action* involves selecting an action schema and running its associated controller. *Prediction* involves monitoring the action and, if its trajectory enters a decision region, or is not as predicted, indicating a failure condition and perhaps selecting a new action. *Learning* involves reinforcing sequences of action schemas and forming gists, and, when actions have high-entropy distributions over decision regions, *state-splitting* to produce lower-entropy schemas. These processes are described in detail in the following sections.

[1] http://www.ode.org/

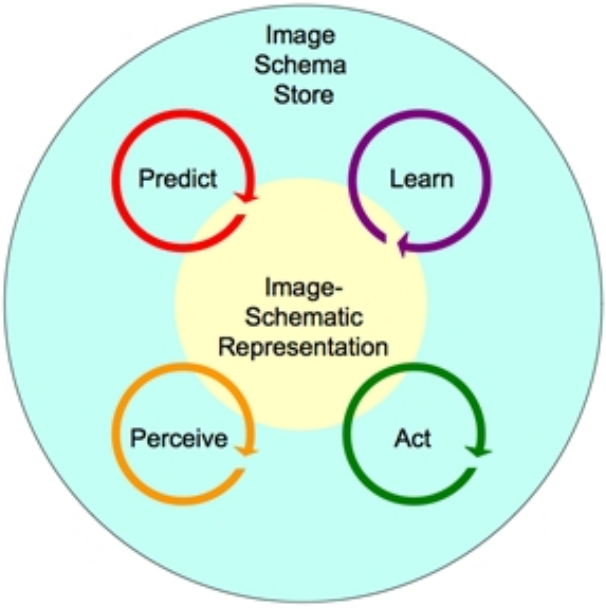

Figure 1: Jean's architecture builds image-schematic representations of the current scenario (perception), selects and implements action schemas (act), predicts how the scenario will unfold (predict), and learns new schemas and gists (learn).

4.1 What is Innate?

As engineers, we must provide enough innate knowledge of the right kind to support learning vast amounts of commonsense knowledge. It follows that the innate endowment should be general — more like the Cyc upper ontology than specialized facts. Why not the appropriate parts of Cyc? The main reason is that Cyc contains logical axioms *about* the world, not representations (or elements thereof) *of* the world. Cyc might serve as a meta-language for reasoning about representations of the world; for example, given a representation of one block on top of another, Cyc presumably could reason about the entailments of "on-ness." What we need, though, is a representational system we can hook up to a sensory system — something that will produce spatial representations that change as objects move — and Cyc isn't that.

Instead, we have developed a representation language called the *Image Schema Language* (ISL; [32]). This effort implicitly says image schemas are innate, so let's look briefly at the psychology and linguistics literature on image schemas.

4.2 Image Schemas

Image schemas are representations that are "close" to perceptual experience. Some authors present them as re-descriptions of experience. Their popularity is due to their supposed generality and naturalness: Many situations are naturally described in terms of paths, up-down relations, part-whole relationships, bounded spaces, and so on. Even non-physical ideas, such as following an argument, containing political fallout, and feeling "up" or "down," seem only a short step from image-schematic foundations [20, 19, 16, 15, 34].

In cognitive linguistics [24], "an image schema is a condensed redescription of perceptual experience for the purpose of mapping spatial structure onto conceptual structure. According to Johnson [16], these patterns 'emerge as meaningful structures for us chiefly at the level of our bodily movements through space, our manipulations of objects, and our perceptual interaction'." Image schemas have also been suggested to play a critical developmental role, forming the basis of early cognitive development, and possibly extending to all sensori-motor perceptual modalities [21, 22].

Almost all image-schematic "theories" are post-hoc. Some steps have been taken toward the computational formalization of image schemas (notably, [2, 29]), but image schemas are still largely discussed in descriptive, qualitative, abstract terms.

5 ISL: An Image Schema Language

Our intention has been to provide some image schemas as "innate" knowledge and learn the rest. One can go mad trying to find the best set of innate schemas (or, indeed, any ontological foundations) so, instead, we decided to implement a set of schemas that have been developed in the cognitive linguistics literature as a foundation for lexical semantics. This set is shown in Table 1. To date, ISL implements roughly half of them.

Unfortunately, there are two problems with the image schemas described in the literature: They are ambiguous, and most of them describe static arrangements. For example, the *path* schema might represent a static arrangement in physical space (e.g., a path between two buildings) or a path one intends to follow (e.g., the same path between buildings but with an intentional gloss) or the path one is actually following (e.g., the moment-by-moment location as one moves between the buildings).

Eventually we developed three kinds of image schemas: Static schemas, which describe instantaneous snapshots; dynamic schemas, which represent non-agentive change; and action schemas, which represent agentive change. These are described in detail in the following sections.

5.1 Implementing Image Schemas

As represented in ISL, image schemas are objects, in the sense of the object-oriented data model. Each schema has a set of operations that determine its capabilities. For example, operations for a *container* schema include putting material into a container and taking material out. Each schema also has a set of internal slots that function as the equivalent of roles in a case grammar sense [12]. Slots permit

Table 1: Image schemas described by Croft and Cruse (2004)[2].

Space:	Location, Up-Down, Front-Back, Left-Right, Near-Far, Verticality, Center-Periphery, Straight, Contact
Force:	Compulsion, Blockage, Diversion, Counterforce, Restraint, Resistance, Attraction, Enablement
Containment:	Container, In-Out, Surface, Content, Full-Empty
Locomotion:	Momentum, Path
Balance:	Axis Balance, Twin-Pan Balance, Point Balance, Equilibrium
Identity:	Matching, Superimposition
Multiplicity:	Merging, Collection, Splitting, Iteration, Part-Whole, Linkage, Count-Mass
Existence:	Removal, Bounded space, Cycle, Object, Process, Agent
Space:	Up-Down, Front-Back, Left-Right, Near-Far, Center-Periphery, Contact
Scale:	Path
Container:	Containment, In-Out, Surface, Full-Empty, Content
Force:	Balance, Counterforce, Compulsion, Restraint, Enablement, Blockage, Diversion, Attraction
Multiplicity:	Merging, Collection, Splitting, Iteration, Part-Whole, Mass-Count, Link
Identity:	Matching, Superimposition
Existence:	Removal, Bounded Space, Cycle, Object, Process

image schemas to be related to each other through their slot values. For example, the contents of a container can be other image schemas.

Image schemas are related by inheritance; for example, we define a container-with-capacity schema — with an additional slot representing capacity — as a specialization of a basic container.

An important aspect of ISL is its use of *interpretation*. In object-oriented terms, interpretation can be thought of as an extended form of delegation. Interpretations map from one or more specifications of a "source" image schema to a "target" schema. For example, we would probably first think to represent a room as a location or bounded space (i.e., a region) image schema, but from a fire marshall's perspective it would be useful to interpret a room as a container with a capacity of some number of people. Interpretation gives us flexibility in evaluating the properties of some domain in terms of image schemas; different (even conflicting) interpretations can be maintained at the same time for a single "real" object or relationship. Interpretation is also critical to metaphorical extension and bears relations to analogical mapping (e.g., [14]).

---

[2] Croft, W., and Cruse, D.A. *Cognitive Linguistics*. Cambridge University Press, 2004.

5.2 Static Image Schemas

Static schemas represent spatial configurations. They have no dynamical content. Consider a chess board in which the Black queen has the White king in check. In image schema terms, we say that there exists a path from the queen to the king. In ISL, we generate a path schema, which contains a set of locations, as shown in Figure 2. Representing a path simply as a set of locations gives us generality, but here it's important that the queen can traverse the path in the situation that holds currently on the board. This is captured by an interpretation of the path as a set of directional linkages from each location (a source) to the next on the path (a destination). Another piece of domain information is that no location can be occupied by more than one piece at a time. This is represented by an interpretation of each location as a container with a capacity of 1. When a piece moves to a location, the container reaches capacity and yet another image schema, empty/full, is automatically created, indicating that the location is full.

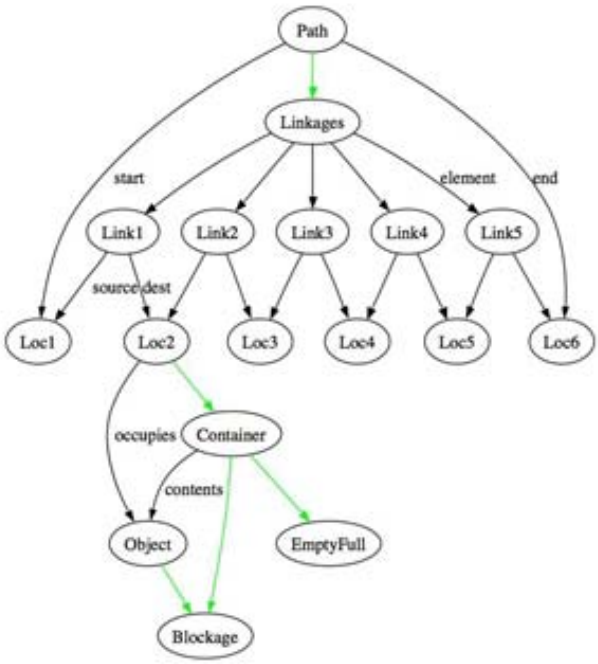

Figure 2: Representing blockage in ISL.

Given these image schemas, their relationships, and the operations that they support, it becomes possible to reason about the situation and the possible responses White can make to counter the threat of the queen. The check exists because the path from the queen to the king is traversable. Traversability for a path schema is defined, in words, as follows: a path can be traversed when every linkage between successive locations can be traversed. Traversability for a linkage schema, in turn, is allowed when its source can be entered and its destination can be exited. Basic locations have no built-in constraints on entering and exiting, but when a location is interpretable as a container, this changes. One cannot add more to a container that has reached capacity. The interpretation relationships between these schemas cause changes to propagate outward: a full container cannot be added to; its location cannot be entered; a directional linkage cannot be traversed (via its source); a path cannot be traversed (due to a non-traversable linkage). The result is a new image schema, blockage, which is created when a container representing a location that

acts as the source of a directional linkage in a path becomes full. The contents of the container constitute the blocker. This structured combination of image schemas—locations, path, linkages, blockage, and so forth—can be stored away in memory for later retrieval, limiting the need for a complete reconstruction of the combination from scratch.

The ISL representation provides a description of the situation in the form of a structured combination of image schemas. Compare this combination with how we might describe a tactic in chess: "When an opponent's piece puts your king in check, you can counter by moving another piece into its path." The combination of schemas captures the essence of this natural language description. The representation is general, abstracting away the specific positions of the pieces, the existence of other pieces, even the identity of the attacking piece. The generality of the representation can also be seen in that its substructure maps to other basic concepts in chess. By using object schemas that include information about the color of a piece, we can use the path/linkage substructure to represent a threat of one piece on another, when the colors of the pieces are different; if they are the same, we can represent a defense relationship. The representation also supports the ability to reason about emergent structure. White might have a dozen possible moves in the situation given in the example, but few of them will be appropriate (or even legal). One of White's most plausible responses, in terms of image schemas, is to recognize that the situation is a partial match to a blockage schema (which does not yet exist), and that a specific response will lead to the creation of the blockage. Rather than reasoning about the low-level properties of individual pieces, White reasons using tactical abstractions. Other chess concepts similarly lend themselves to abstraction that can be naturally captured by image schemas: application of force on the opponent's king (even if the king is never put in check), balance in the distribution of pieces on the board, control of the center of the board, and so forth. Lower-level descriptions of moves (e.g., based on paths alone) are not inaccurate, but they fail to capture the reasons behind the moves.

5.3 Action Schemas

Action schemas have three components: controllers, maps, and decision regions. Controllers control Jean's behavior. For example, the (move-to Jean Obj) controller moves Jean from its current location to the specified object. Jean has very few innate controllers — move-to, turn, rest, apply force. It learns to assemble controllers into larger plan-like structures called *gists* as described in Section 6.1.

As Jean moves (or executes any other controller) certain variables change their values; for instance, the distance between Jean's current location and Obj usually decreases when Jean executes the move-to controller. Similarly, Jean's velocity will typically ramp up, remain at a roughly constant level, then ramp down as Jean moves to a location. The values of these variables ground out in Jean's sensors, although some variables correspond to processed rather than raw sensory information. In particular, some variables in maps may correspond to the presence or absence, or level, or variables in static image schemas; for instance, the presence or absence of an object, or the distance one object is behind another.

These variables serve as the dimensions of maps. Each execution of a particular schema produces a trajectory through a map — a point that moves, moment by moment, through a space defined by distance and velocity, or other bundles of variables. Each map is defined by the variables it tracks (typically, several variables), and different maps will be relevant to different controllers.

Each invocation of a controller creates one trajectory through the corresponding map, so multiple invocations — repeating an action several times — will create multiple trajectories. Figure 3 shows several such trajectories for a map of distance for the move-to controller. One can see that move-to has a "typical" trajectory, which might be obtained by taking the mean of the trajectories in the map. Also, when operating in Jean's environment, move-to produces trajectories that lie within some distance of the mean trajectory. In a manner reminiscent of Quality Control Jean can assess whether a particular trajectory is "going out of bounds."

Every map has one or more *decision regions* within which Jean may decide to switch from one controller to another. One kind of decision region corresponds with achieving a goal; for example, there is a decision region of the move-to map in which distance to the desired location is effectively zero (e.g., the thin, horizontal grey region in Figure 4). Another kind of decision region corresponds to going out of bounds and being unable to achieve a goal; for instance, there is a region of a time-distance map from which Jean cannot move to a desired location by a desired time without exceeding some maximum velocity (e.g., the inverted wedge-shaped region in Figure 4). These regions are sometimes called *envelopes*, [13, 7, 1, 28]

Jean is not the only agent in its environment and some maps describe how relationships between Jean and other agents change. The upper two panels of Figure 6 illustrates how distance, relative velocity, heading, and contact change in an environment that includes Jean and another agent, called the "cat," an automaton that moves away from Jean if Jean moves too close, too quickly.

The idea that sensori-motor and pre-operational development should rely on building and splitting maps or related dynamical representations was anticipated by Thelen and Smith [35] and has been explored by other researchers in developmental robotics and psychology (e.g., [30, 9, 31, 3, 21, 22]).

5.3.1 Action Schemas as Finite State Machines

It will help to draw parallels between Jean's maps and the more familiar elements of finite state machines (FSMs). Conventionally, states in FSMs represent static configurations (e.g., the cat is asleep in the corner) and arcs between states represent actions (e.g., (move-to Jean cat)). For us, arcs correspond to the intervals during which Jean executes controllers (i.e., actions), and states correspond to decision regions of maps. That is, the elements of action schemas are divided into the intervals during which a controller is "in bounds" (the arcs in FSMs) and the intervals during which Jean is thinking about what to do next (the states). Both take time, and so require a rethink of the conventional view that states in FSMs persist and transitions over arcs are instantaneous. However, the probabilistic semantics of FSMs are retained: A controller invoked from a decision region (i.e., an action invoked in a state) will generally take Jean into one of several decision regions (i.e., states), each with some probability. Figure 5 redraws Figure 4 as a finite state machine. Starting from a decision region of some action schema A, the move-to controller will, with some probability (say .7), drop Jean in the decision region associated with achieving its goal and, with the complementary probability, in the region associated with being unable to achieve the goal in time.

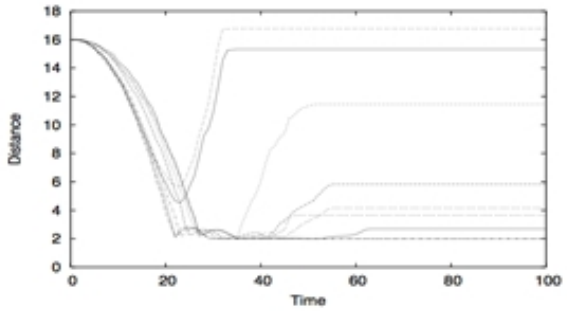

Figure 3: Trajectories through a map of distance between Jean and a simulated cat. Sometimes, when Jean gets close to the cat, the cat moves away and the distance between them increases, in which case a trajectory may go "out of bounds."

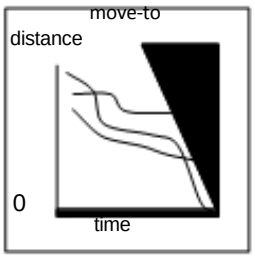

Figure 4: This schematic of a map, for the (move-to Jean Obj) controller, has one decision region associated with the distance between Jean and the object Obj being zero. The other decision region bounds the area in which Jean cannot reach loc even moving at maximum speed.

5.3.2 Dynamic Schemas

There is one special case: Sometimes the world changes when Jean is doing nothing. We model this as a schema in which no controller is specified, called a *dynamic schema* to distinguish it from an action schema. Dynamic schemas do have maps, because variables such as distance between Jean and another agent can change even when Jean does nothing; and they have decision regions, because Jean may want to invoke a controller when these variables take particular values (e.g., moving away when another agent gets too close). The FSMs that correspond to dynamic schemas have no controller names associated with arcs but are otherwise as shown in Figure 5.

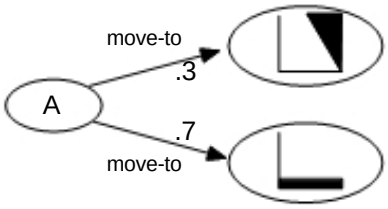

Figure 5: The action schema from Figure 4 redrawn as a finite state machine in which arcs correspond to the execution of controllers and states correspond to decision regions.

6 Learning Actions Schemas and Gists

Jean learns new action schemas and gists, and, quite recently, Jean has started to learn the meanings of words. This section describes the Experimental State Splitting algorithm for learning action schemas and gists. It also reports some experimental results. Section 8 is about lexical learning.

6.1 Experimental State Splitting

Jean learns new schemas in two ways, by *composing schemas* into gists and by *differentiating* states in action schemas. Both are accomplished by the Experimental State Splitting (ESS) algorithm. The basic idea behind Experimental State Splitting (ESS) is simple. The algorithm starts with a minimal state model of the world, in which it has only one all-encompassing state. This model is modified as the agent (i.e., Jean) explores its world, so that it becomes more predictive of some measure of observed action outcomes.

For a general developmental account we want a general measure, not a task-specific one. To accord with the idea that learning is itself rewarding, this measure might have something to do with the informativeness or novelty or predictability of states. In Jean, the ESS algorithm uses a measure we call *boundary entropy*, which is the entropy of the next state given the current state and an intended action. Initially, when there is only one state in the model, the entropy will always be zero. One way to drive Jean toward more states, and, thus toward states that have boundary entropy, is to have a goal state in addition to the initial (non-goal) state. At any moment in time, the agent is in one of these two states, and each state-action pair generates some probability distribution over the set of possible next states. ESS calculates the entropy of this distribution and uses it as a state splitting criterion.

In general, Jean is driven by ESS to modify its world model by augmenting existing states with new states that reduce the boundary entropies of state-action pairs. This augmentation is achieved by splitting an old state into two (or more) new states based on distinguishing characteristics. The state machine thus grows over time as the agent adds more attributes to its state descriptions.

As Jean interacts with the world, it counts the transitions between the states via particular actions. If the developing state machine model is Markov, then the model is a Markov Decision Process (MDP), which Jean can solve for the optimal policy to reach its goal state. In this way, the developing model can be used for planning. However, it is worth pointing out that since Jean operates in a continuous environment using fairly general action schemas, its world model is more like a semi-Markov Decision Process (SMDP), where each action results in a transition between states after a certain amount of time, and this time interval is drawn from some probability distribution.

If Jean lives always in one environment with one set of goals, then ESS will eventually produce optimal policies for the environment. However, the purpose of the Jean project is not to produce optimal policies for each task and variant of Jean's environment, but, rather, to explain how a relatively small set of policies may quickly *accommodate* (as Piaget called it) or *transfer* to new tasks and environments. Our approach to this problem is to extract *gists* from policies. Gists are like policies, in that they tell Jean what to do in different situations, but they are more general because they extract the "usual storyline" or essential aspects of policies. We claim that these essential aspects are typically the *causal relationships* that govern actions and effects in the environment. If an agent can identify and learn these causal relationships, then it should have a very good idea of how its actions affect the world and how act to achieve its goals, *even in novel situations*.

We give a formal outline of the Experimental State Splitting (ESS) algorithm in this section. Jean receives a vector of features $F^t = \{f_1, \ldots, f_n\}$ from the environment at every time tick t. Some features will be map variables, others will be inputs that have not yet been associated with maps. Jean is initialized with a goal state s_g and a non-goal state s_0. $\mathcal{S}_t$ is the entire state space at time t. A is the set of all controllers, and $A(s) \subseteq A$ are the controllers that are executed in state $s \in \mathcal{S}$. Typically $A(s)$ should be much smaller than A. $H(s_i, a_j)$ is the boundary entropy of the state s_i in which controller a_i is executed. A small boundary entropy corresponds to a situation where executing controller a_j from state s_i is highly predictive of the next observed state. Finally, $p(s_i, a_j, s_k)$ is the probability that executing controller a_j in state s_i will lead to state s_k.

For simplicity, we will focus on the version of ESS that only splits states; an alternative version of ESS is also capable of learning specializations of parameterized controllers. The ESS algorithm follows:

- Initialize state space with two states, $\mathcal{S}_0 = \{s_0, s_g\}$.
- While ϵ-optimal policy not found:
  - Gather experience for some time interval τ to estimate the transition probabilities $p(s_i, a_j, s_k)$.
  - Find a schema feature $f \in F$, a threshold $\theta \in \Theta$, and a state $s_i \in \mathcal{S}$ to split that maximizes the boundary entropy score reduction of the split: $\max_{\mathcal{S},A,F,\Theta} H(s_i, a_i) - \min(H(s_{k1}, a_i), H(s_{k2}, a_i))$, where s_{k1} and s_{k2} result from splitting s_i using feature f and threshold θ: $s_{k1} = \{s \in s_i | f < \theta\}$ and $s_{k2} = \{s \in s_i | f \geq \theta\}$.
  - Split $s_i \in \mathcal{S}_t$ into s_{k1} and s_{k2}, and replace s_i with new states in $\mathcal{S}_{t+1}$.
  - Re-solve for optimal plan according to p and $\mathcal{S}_{t+1}$

Finding a feature f and the value on which to split states is equivalent to finding a decision region to bind a map.

Without heuristics to reduce the effort, the splitting procedure would iterate through all state-controller pairs, all features $f \in F$, and all possible thresholds in Θ, and test each such potential split by calculating a reduction in boundary entropy. This is clearly an expensive procedure.

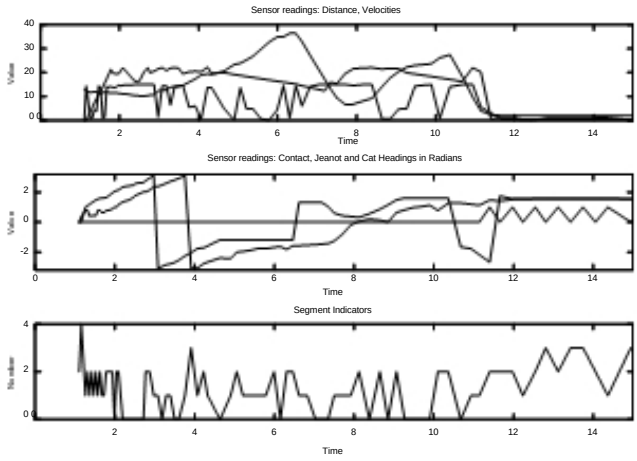

Figure 6: New states are indicated when multiple state variables change simultaneously.

ESS uses a simple heuristic to find threshold values for features f and, thus to split a state: States change when several state variables change more or less simultaneously. This heuristic is illustrated in Figure 6. The upper two graphs show time series of five state variables: headings for Jean and the cat (in radians), distance between Jean and the cat, and their respective velocities. The bottom graph shows the number of state variables that change value (by a set amount) at each tick. When the number of state variables that change simultaneously exceeds a threshold, Jean concludes that the state has changed. The value of the schema f at the moment of the state change is likely to be a good threshold for splitting f. For example, between time period 6.2 and 8, Jean is approaching the cat, and the heuristic identifies this period as one state. Then, at time period 8, several indicators change at once, and the heuristic indicates Jean is in a new state, one that corresponds to the cat moving away from Jean.

6.2 An Example

Figure 7 illustrates a gist for approaching and contacting a simulated cat. In the scenario in which this gist was learned, the cat is animate, capable of sitting still, walking or running away. The cat responds to Jean. In particular, if Jean moves toward the cat rapidly, the cat will run away; if Jean approaches

slowly, the cat will tend to keep doing what it is doing. Because of these programmed behaviors, there is uncertainty in Jean's representation of what the cat will do, but there is a general rule about how to catch the cat, and it can be represented in a gist: The only way to catch the cat is to first get into state s_2 (Figure 7), where the cat is nearby and not moving quickly, and then to move fast toward the cat, reaching state s_1. All other patterns of movement leave Jean in states s_3 or s_4. This corresponds to the strategy of slowly sneaking up to the cat and then quickly pouncing on it to catch it.

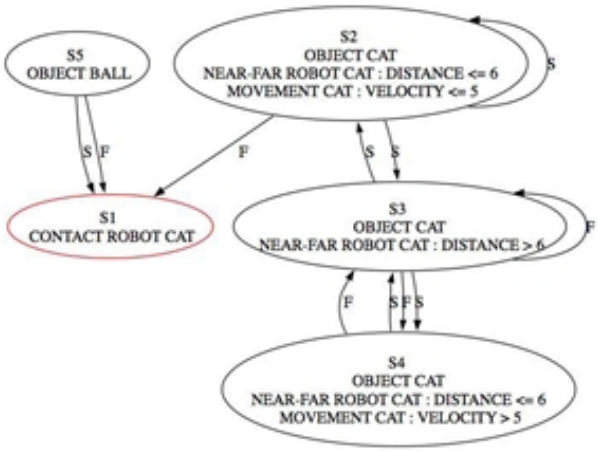

Figure 7: A learned composite action schema for catching a cat.

To learn a gist like the one in Figure 7, Jean repeatedly retrieves action schemas from its memory, runs the associated controllers, producing actions, specifically slow and fast movement to a location; assesses the resulting states, and, if the transitions between states are highly unpredictable, Jean splits states to make the resulting states more predictable.

In fact, the three states, s_2 , s_3 , and s_4 were all originally one undifferentiated state in which Jean moved either fast or slowly toward the cat. Jean's learning history — the distinctions it makes when it splits states — begins with a single, undifferentiated non-goal state. Then, Jean learns that the type of object is an important predictor of whether or not it can catch the object. Balls are easy to catch, whereas cats are hard to catch. From here, ESS recognizes that distance also influences whether or not it can catch a cat. Starting near the cat, a fast-approach-object (*F*) action will often catch the cat, whereas this action will not usually catch the cat from further away. Thus, ESS splits on distance with a threshold of 6, where <= 6 is considered near, and > 6 is far. Finally, ESS may notice that even when Jean is near the cat, sometimes it does not succeed in catching the cat. This might be because the cat is already moving away from the agent with some speed. Thus, ESS may do a final split based on the velocity of the cat. This process leads to the states s_2 , s_3 , s_4 and s_5 that we see in Figure 7.

7 Transferring Learning

Although ESS can learn action schemas and gists for new situations from scratch, we are much more interested in how previously learned policies can accommodate or transfer to new situations. This is, after all, how children learn. Gists capture the most relevant states and actions for accomplishing past goals. It follows that gists may be transferred to situations where Jean has similar goals and the conditions in the situation are similar.

The version of ESS that we described above is easily modified to facilitate one sort of transfer: After each split we remove the transition probabilities on all action transitions between each state. This allows the state machine to accommodate new experience while maintaining much of the structure of the machine (see [18] for a previous example of this idea). In the experiments in the next section we explore the effects of transfer using this mechanism in several conditions.

7.1 Experiments

To measure transfer we adopt a protocol sometimes called **B/AB**: In the **B** condition the learner learns to perform some tasks in situation or scenario *B*. In the **AB** condition, the learner first learns to perform tasks in situation or context *A* and then in *B*. By comparing performance in situation *B* in the two conditions after different amounts of learning in situation *B* one can estimate the effect of learning in *A* and thus the knowledge transferred from situation *A* to situation *B*. In our experiments, better learning performance means less time to learn a gist to perform a task at a criterion level. Thus, a smaller area beneath the learning curve indicates better learning performance, and we compare conditions **B** and **AB** by comparing the area beneath the learning curves in the respective conditions.

We tested Jean's transfer of gists between situations in the 3-D real time strategy game platform ISIS. ISIS can be configured to simulate a wide variety of military scenarios with parameters for specifying different terrain types, unit types, a variety of weapon types, and multiple levels of unit control (from individual soldiers to squad-level formations).

In each of three experiments, Jean controlled a single squad at the squad level, with another squad controlled by an automated but non-learning opponent. Jean's squad ranged in size from 7 to 10 units, while the opponent force ranged from 1-3 units. Although the opponent was smaller, it could move faster than Jean's forces. In each experiment, Jean's goal is to move its units to engage and kill the opponent force. Each experiment had a transfer condition **AB** and a control condition, **B**. In the former, Jean learned gists to accomplish its goal in a scenario designated *A* and then learned to accomplish its goal in scenario *B*. In the latter, control condition, Jean tried to learn in scenario *B* without benefit of learning in *A*.

Jean is provided four innate action schemas: `run`, `crawl`, `move-lateral`, and `stop-and-fire`. It must learn to compose these into gists that are appropriate for different engagement ranges, possible entrenchment of the opponent, and some terrain features (mountains).

The experiments differ in their *A* and *B* scenarios:

Experiment 1: All action takes place in open terrain. *A* scenarios all have Jean's forces starting near enemy forces. *B* scenarios are an equal mix of starting near the enemy or far away from the enemy.

Experiment 2: All action takes place in open terrain. *A* scenarios all have Jean's forces starting far from the enemy forces. *B* scenarios are an equal mix of starting near the enemy or far away from the enemy.

Experiment 3: The terrain for *A* scenarios is open, whereas the terrain for *B* scenarios has a mountain that, for some placements of Jean's and the enemy's forces, prevents them seeing each other. (The advantage goes to Jean, however, because Jean knows the location of the enemy forces.) The *A* scenario is an equal mix of starting near or far from the enemy, the *B* scenario is an equal mix of starting near and far from the enemy in the mountain terrain.

7.2 Metrics and Analysis

We plot the performance of the Jean system in the various experimental scenarios as learning curves over training trials. Better learning performance is indicated by a smaller number of training instances required by Jean to achieve a criterion level of performance. Thus, a smaller area beneath the learning curve indicates better learning. We developed a randomized-bootstrap test of significance to test the hypothesis that Jean transfered knowledge from condition A to B (see [8] for details of the analysis).

7.3 Results

Let us start with a qualitative assessment of what Jean learned. In Experiment 1, Jean learned in scenario *A* to run at the enemy and kill them. In scenario *B*, Jean learned a gist that included a conditional: When starting near the enemy, use the gist from scenario *A*, but, when starting far from the enemy, crawl — don't run — until one is near the enemy and then use the gist from scenario *A*. The alternative, running at the enemy from a far starting location, alerts the enemy and causes them to run away. State splitting did what it was supposed to do: Initially, Jean's gist for scenario *B* was its gist for *A*, so Jean would always run at the enemy, regardless of starting location. But through the action of state splitting, Jean eventually learned to split the state in which it ran at the enemy into a run-at and a crawl-toward state, and it successfully identified the decision region for each. For instance, the decision region for the crawl-toward state identifies a distance (corresponding to being near the enemy), from which Jean makes a transition to the state in which it runs at and shoots the enemy.

Similar results are obtained in Experiment 3, where Jean learns to run at the enemy from a far starting location as long as the mountain prevents the enemy from sighting Jean, otherwise to crawl.

In Experiment 2, Jean learned nothing in scenario *A* and was no more successful in scenario *B*. This is due to the difficulty of the scenario. Jean is always initialized far away from the enemy units, and must learn a policy for killing them by exploring a continuous, high-dimensional feature space using her four available actions. Many of these actions result in the enemy soldiers detecting Jean's presence and running away, thus reducing Jean's chances of ever reaching her goal by simple exploration. Since Jean does not learn anything useful in the *A* scenario, her performance in the **AB** transfer condition is no better than in the control condition **B**.

8 Learning Word Meanings

A recent version of Jean is situated in a simulated physical room full of blocks. Jean can move around the room and pick up and carry the blocks. This environment, called *Jean's Room*, is a testbed for learning the meanings of words. A human participant instructs Jean in English, as shown in Figure 8. For instance, he might type, "Pick up the green cone," and Jean might reply, "which is the green cone." When the human identifies it, Jean has the opportunity to learn what the words "green" and "cone" refer to, or to refine earlier hypotheses about the meanings of these words.

The learning method is quite simple and intuitive: Each word is represented by a vector of features and feature values, maintained in a large table (the orange block in Figure 8). When Jean is presented with a scene, it produces an image-schematic representation (the yellow block in Figure 8). When presented with a sentence such as "Put the blue one behind the green cone," Jean runs it through a parser, and then associates words in the sentence with aspects of the scene, constrained by the parse. For example, Jean knows there are two objects in the scene, one of which it is holding (or containing, in image-schematic terms). Jean knows (in-front-of Obj-1 Obj-2) and, conversely, (behind Obj-2 Obj-1). Suppose the word "behind" is not known to Jean. Because Jean has parsed the sentence, it knows that "behind" is a preposition, so it guesses that "behind" refers to either the in-front-of or the behind relationship between the objects. Jean updates the score of each of these hypotheses in the word/model feature table using a technique called regret minimization. Over time, model features become associated with words, so in-front-of becomes associated with "front," and behind becomes associated with "rear," and "back," and so on.

This associative method is closely related to previous work we have done in lexical semantics (e.g., [36, 25,27,10]).

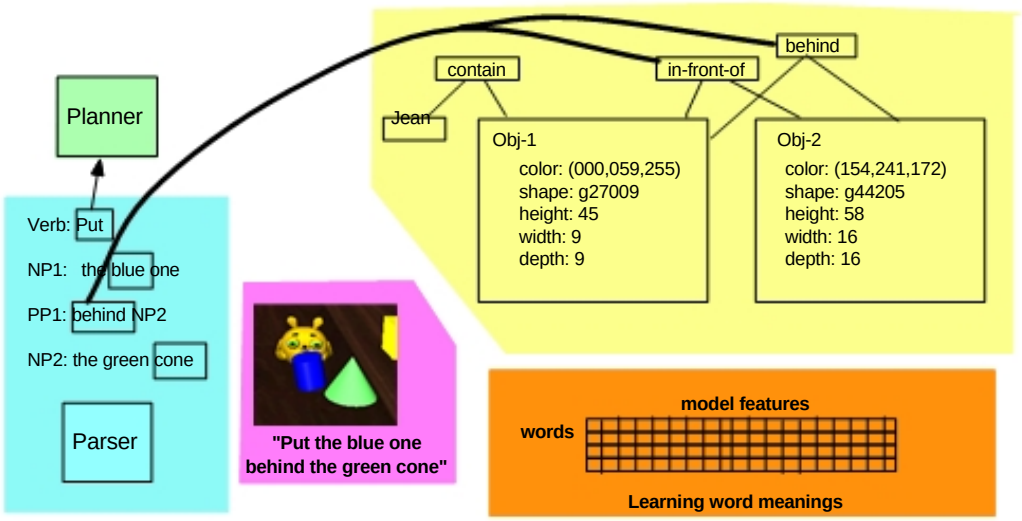

Figure 8: Jean learns word meanings by building an image-schematic model of the scene (pale yellow box), parsing the sentence (pale blue box), and probabilistically associating words with aspects of the scene in a large table (orange box).

9 Future Work

The map-based representation of action schemas has been extended in several ways in our lab, though the techniques have not yet been incorporated into the Jean project. Maps provide a natural representation for the lexical semantics of some verbs. Indeed, verbs that differ subtly in *manner* can be differentiated in the map formalism, though not easily in a logical formalism. For instance, it is straightforward to differentiate *shove* from *push* with maps. Another advantage of maps is that their regions can serve as goals for the purpose of planning, and also for interpreting intentions. These aspects of maps are described in the following sections.

9.1 Representing the Meanings of Verbs

Previous work has shown that robots can learn the meanings of words by associating aspects of the perceptual array with utterances (e.g., [33, 25, 31, 6, 10]). For a comprehensive review of the psychological literature on dynamics and word meanings see [4]. A persistent question in the work is whether the perceptual array contains enough information to provide semantics for words, or, in a slightly different formulation, what fraction of the variability of word use is explained by information in the perceptual array? A concrete version of this question is posed here: What fraction of the variability of word use is explained by the dynamical aspects of interactions between two objects?

In [9], we developed 18 movies of interactions between object. Each movie was generating by a program written in *breve* [17], an animation tool with good physics. A vector of parameters of each program characterizes the dynamics of the corresponding movie. We showed the movies to pre-school children and asked them to describe the action in the movies. After removing non-content words, we characterized each movie by a distribution of words. We ranked all pairs of movies in two ways: by the similarities of their word distributions and by the similarities of their vectors of program parameters. Then we compared the rankings. The results are highly significant: there is strong dependence between the parameters of the programs that generated the movies and the distributions of words that children use to describe the movies.

9.1.1 Maps for Verbs

In the *maps for verbs* representation of verb meanings, the denotations of verbs dealing with interactions between two bodies, such as push, hit, chase, and so on, are represented as pathways through a metric space, or map, the axes of which are perceived distance, velocity, and energy transfer [6]. Verbs with similar meanings have similar pathways. A scene, such as one object chasing another, is thought to be perceived as a pathway through the map. To learn verb meanings, one simply associates verbs that describe scenes with the corresponding pathways.

Although maps are compact and objective representations of some verb meanings, we do not know whether they have psychological reality — whether humans use maps to assign meanings to verbs. Even if they do, the original maps for verbs representation might have the wrong axes, or the axes might be correct but verbs might not be correlated with the particular features of pathways, as we thought. The experiment in this paper does not test whether humans have maps in their heads. Instead it asks, “If one creates movies which are different according to the maps for verbs framework, will human subjects use different distributions of words to describe them?”

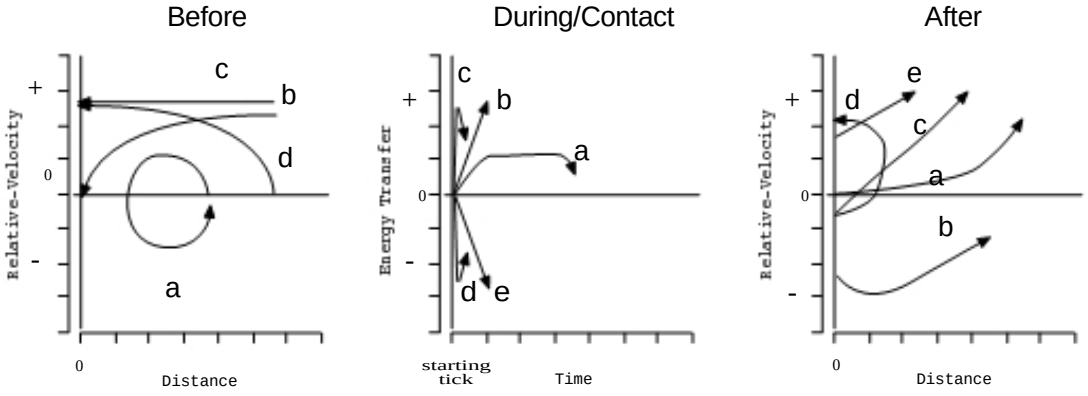

Figure 9: Maps-for-verbs model of the three phases of interaction.

In the maps for verbs framework, the dynamics of interaction are split into before, during (contact), and after phases. Figure 9 depicts these phases with illustrative trajectories in each. The axes of the maps are the

same in the *before* and *after* phases: they are relative velocity and distance between the two bodies. Relative velocity is the difference between the velocity of one body, A, and another, B: *Velocity* (*A*) − *Velocity* (*B*). Many verbs (e.g., transitive verbs) predicate one body as the "actor" and the other as the "target" (or "subject" or "recipient") of the action. For example, in a push interaction, the actor does the pushing, and the target is the body being pushed. By convention, the actor is designated as A and the target is B. Thus, when relative velocity is positive, the actor's velocity is greater than that of the target; and when relative velocity is negative, the target's velocity is greater than that of the actor. Distance, in turn, is simple Euclidean distance between the bodies.

The vertical dimension of the map in the *during* phase is perceived energy transfer (from the actor to the target). If energy transfer is positive, then the actor is imparting to the target more energy than the target originally had; if energy transfer is negative, then the situation is reverse and the target is imparting more energy to the actor. Since energy transfer is not directly perceivable, we approximate it by calculating the acceleration of the actor in the direction of the target while the actor and target are in contact.

The labeled trajectories in Figure 9 characterize the component phases of seven interaction types, as described by the verbs push, shove, hit, harass, bounce, counter-shove and chase.

For example, 〈**b, b, b**〉 describes a *shove*. The actor approaches the target at a greater velocity than the target, closing the distance between the two bodies. As it nears the target, the actor slows, decreasing its velocity to match that of the target. Trajectory **b** of the before phase in Figure 9 illustrates these dynamics, showing the decrease in relative velocity, along with decrease in distance. At contact, the relative velocity is near or equal to zero. During the contact phase, the actor rapidly imparts more energy to the target in a short amount of time, as illustrated by **b** of the during/contact phase. And after breaking-off contact with the target, the agent rapidly decreases its velocity while the target moves at a greater velocity due to the energy imparted it.

With this three-phase representation scheme, we define six more interaction types corresponding to common English verbs:

- *Push* 〈**b, a, a**〉 - begins like shove, but at contact relative velocity is near or equal to zero and the actor smoothly imparts more energy to the target; after breaking contact, the agent gradually decreases its velocity.

- *Hit* 〈**c/d, c, c**〉 - may begin with the actor already at high velocity relative to the target or increasing in relative velocity, and thus is characterized by **c** or **d** in the before phase.

- *Harass* 〈**c/d, c, d**〉 - is similar to a hit, except the after-phase involves the actor quickly recovering its speed and moving back toward the target, not allowing the distance between the two to get very large (the **d** in the after phase). Harass highlights that interactions may be cyclic: the after phase of one epoch blends into the before phase of the next.

- *Bounce* 〈**c/d, d, e**〉 - along with counter-shove, bounce involves the target making a more reactive response to the actor's actions. Bounce begins like a hit or harass, but at contact, the target transfers a large amount of energy back to the actor.

• *Counter-shove* 〈**b/c/d, e, e**〉 - is a version of a shove where the target imparts energy to the actor.

• *Chase* 〈**a**, −, −〉 - involves the actor moving toward the target, closing the distance between the two, but never quite making contact, so the during and after phases are not relevant. This is depicted as the circular trajectory **a** in the before phase.

9.2 Maps and Intentions

Suppose at a party a drunk man pats you on the back a little too hard, knocking you forward. Is this a pat gone awry, or a not-too-subtle aggression? You don't know. You don't know his intention. Figure 10 shows two representations of the interaction. The actual trajectory is the same in both: His hand makes contact with your back at a relative velocity greater than zero, and it transfers a considerable amount of energy to your back. The difference between the representations is the mans goal regions. On the left, you see a benign pat gone wrong. The goal region for relative velocity (the shaded area in the before phase) is considerably lower than the one the man actually generated (he's drunk, after all). The trajectory for energy transfer and displacement of your back falls well outside his goal region, also. On the right, however, the man generates the relative velocity profile that he intends, and he hits you as hard as he intends, and you are knocked forward as far as he intends.

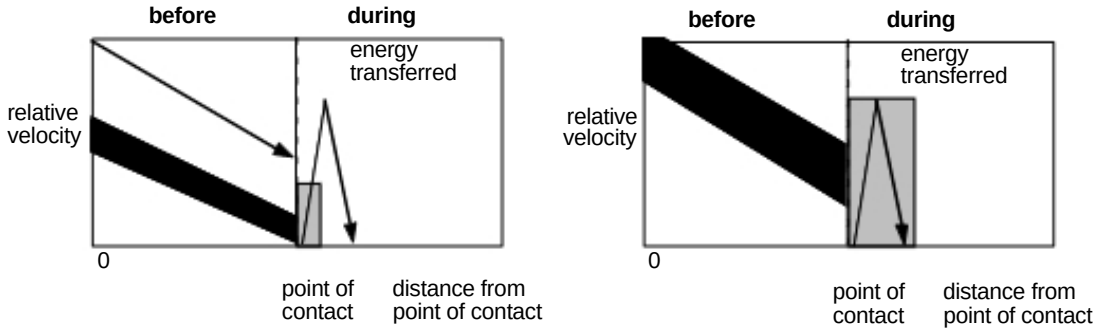

Figure 10: Intentions can be represented as regions of maps.

Generally, you don't know the intentions of other parties, so you cannot be sure that "hit" is the correct verb to describe an interaction like this. But sometimes, dynamics alone are sufficient to infer intent, albeit heuristically. If the man doesn't intend to hurt you, then he will probably try to modulate his arm movement when he realizes it is too fast. But what if he actually increases his arm speed as he approaches you? Then it requires uncommon charity to excuse his behavior as accidental.

Jean will eventually be able to infer the intentions of others by imagining the regions of their maps that they intend to occupy.

9.3 Maps and Planning

While it is sometimes difficult to infer the intentions of others, one usually knows one's own. So when one engages in an activity, one has a map in mind with normative regions that represent goals, failures, and warnings of failures. Recalling that maps can be re-represented as finite state machines, such as Figure 7, we can imagine planning maps with a Partially Observable Markov Decision Process (POMDP) method or something similar. A more psychologically plausible method is means-ends analysis to generate sequences of actions, which are then learned as gists. Means-ends analysis is well-known to underlie a vast range of human problem solving behavior [23]. It requires a model of the effects of actions called an *operator-difference table*. In several earlier projects we learned operator-difference tables, based on maps, for a mobile robot [26, 11]. We are eager to reprise this work in the Jean project.

References

[1] Atkin, M. S., and Cohen, P. R. Searching continuous state spaces effectively using critical points. Submitted to the *Sixteenth National Conference on Artificial Intelligence*, 1999.

[2] Bailey, D. Getting a grip: A computational model of the acquisition of verb semantics for hand actions, Feb. 04 2000.

[3] Barsalou, L. W. Perceptual symbol systems. *Behavior and Brain Sciences*, 22, 1999, 577-609.

[4] Cannon, E., and Cohen, P. *Word choice as a conditional probability: Motion-based semantics*. Oxford University Press, New York, NY, 2005.

[5] Chang, Y., Morrison, C. T., Kerr, W., Galstyan, A., Cohen, P. R., Beal, C., St. Amant, R., and Oates, T. The jean system. In *Proceedings of the Fifth International Conference on Development and Learning (ICDL 2006)*, 2006.

[6] Cohen, P. R. Maps for verbs. In *Proceedings of the Information and Technology Systems Conference, Fifteenth IFIP World Computer Conference*, 1998.

[7] Cohen, P. R., St. Amant, R. S., and Hart, D. M. Early warnings of plan failure, false positives and envelopes: Experiments and a model. In *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society,* 1992, Lawrence Erlbaum Associates, Inc., pp. 773-778.

[8] Cohen, P. R., Chang, Y.-H., Morrison, C., and Beal, C. R. Learning and transferring action schemas. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI 2007)*, 2007.

[9] Cohen, P. R., Morrison, C. T., and Cannon, E. Maps for verbs: The relation between interaction dynamics and verb use. In *Proceedings of the Nineteenth International Conference on Artificial Intelligence (IJCAI 2005)*, 2005.

[10] Cohen, P. R., and Oates, T. A dynamical basis for the semantic content of verbs. In *Grounding of Word Meaning: Data & Models Workshop (AAAI-98)*, 1998, pp. 5-8.

[11] Cohen, P. R., Sutton, C., and Burns, B. Learning effects of robot activities using temporal associations. In *The 2nd International Conference on Development and Learning (ICDL 2002),* 2002.

[12] Fillmore, C. The case for case. In *Universals in Linguistic Theory*, E. Bach and R. T. Harms (Eds.), Holt, Rinehart & Winston, London, 1968.

[13] Gardiol, N. H., and Kaelbling, L. P. Envelope-based planning in relational mdps. In *Advances in Neural Information Processing Systems 16 (NIPS 2003)*, 2003.

[14] Gentner, D., and Markman, A. M. Structure mapping in analogy and similarity. *American Psychologist, 52*, 1997, 45-56.

[15] Gibbs, G. J., and Colston, H. L. The cognitive psychological reality of image schemas and their transforms. *Cognitive Linguistics 6*(4), 1995, 347-378.

[16] Johnson, M. *The Body in the Mind*. University of Chicago Press, 1987.

[17] Klein, J. breve: A 3d simulation environment for the simulation of decentralized systems and artificial life. In *Proceedings of Artificial Life VIII: The 8th International Conference on the Simulation and Synthesis of Living Systems*, 2002.

[18] Krueger, B., Oates, T., Armstrong, T., Cohen, P. R., and Beal, C. Transfer in learning by doing. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2005),* 2005.

[19] Lakoff, G. *Women, Fire, and Dangerous Things*. University of Chicago Press, 1984.

[20] Lakoff, G., and Johnson, M. *Metaphors We Live By*. University of Chicago Press, 1980.

[21] Mandler, J. How to build a baby: Ii. conceptual primitives. *Psychological Review, 99*, 1992, 597-604.

[22] Mandler, J. *The Foundations of Mind: Origins of Conceptual Thought*. Oxford University Press, 2004.

[23] Newell, A., and Simon, H. *Human Problem Solving*. Prentice-Hall, Englewood Cliffs, NJ, 1972.

[24] Oakley, T. Image schema. In *Handbook of Cognitive Linguistics*, D. Geeraerts and H. Cuyckens, (Eds.). Oxford University Press, 2006.

[25] Oates, J. T. *Grounding Knowledge in Sensors: Unsupervised Learning for Language and Planning*. PhD thesis, Department of Computer Science, University of Massachusetts, 2001.

[26] Oates, T., and Cohen, P. R. Searching for planning operators with context-dependent and probabilistic effects. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 1996, pp. 863-868.

[27] P. Cohen and C. R. Beal. Natural semantics for a mobile robot. In *Proceedings of European Conference on Cognitive Science,* 1999.

[28] Powell, G. M., and Cohen, P. R. Operational planning and monitoring with envelopes. In *Proceedings of the IEEE Fifth AI Systems in Government Conference*, (Washington, DC), 1990.

[29] Regier, T. *The Human Semantic Potential: Spatial Language and Constrained Connectionism*. The MIT Press, Cambridge, MA, 1996.

[30] Rosenstein, M. T., Cohen, P. R., Schmill, M. D., and Atkin, M. S. Action representation, prediction and concepts. In *AAAI Workshop on Robots, Softbots, Immobots: Theories of Action, Planning and Control,* 1997.

[31] Siskind, J. Grounding lexical semantics of verbs in visual perception using force dynamics and even logic. *Journal of AI Research, 15,* 2001, 31-90.

[32] St. Amant, R., Morrison, C. T., Chang, Y., Cohen, P. R., and Beal, C. An image schema language. In *Proceedings of the 7th International Conference on Cognitive Modelling (ICCM 2006)*, 2006.

[33] Steels, L. *The Talking Heads Experiment: Volume I. Words and Meanings*. Laboratorium, Antwerpen, 1999. This is a museum catalog but is in preparation as a book.

[34] Talmy, L. *Toward a Cognitive Semantics, vol. 1: Conceptual Structuring Systems (Language, Speech and Communication)*. The MIT Press, Cambridge, MA, 2003.

[35] Thelen, E., and Smith, L. *A Dynamic Systems Approach to the Development of Cognition and Action*. The MIT Press, Cambridge, MA, 1994.

[36] Tim Oates, Zachary Eyler-Walker, P. R. Cohen. Toward natural language interfaces for robotic agents: Grounding linguistic meaning in sensors. In *Proceedings Fourth International Conference on Autonomous Agents*, 2000, 227-228.